1. Given an array of size 16, what is the maximum possible *mixed-up-ness* score? Explain why you think this (e.g., give a logical argument or provide an example)

when this array is in descending order: 16, 15…1, the number of reverse order: 15+14+…+1=120

2. What is the worst-case runtime of the brute-force algorithm that you designed? Give a proof (a convincing argument) of this.

Design a brute-force algorithm:

for each element within the array, compare it with all elements on its right side and count the number of reverse order pairs. double loop, outer loop is for each element, inner loop is to compare with right side elements and count. O(N^2)

Design a divide-and-conquer algorithm:

1.divide the array to two halves with same size / almost same size. base case is only one element is inside either half

2.like merge sort's merging step, during this step count the number of reverse order(let i be index for the left half and j be index for the right half, if a[i]>a[j], count+=mid-i since both left and right half are sorted).

3.recursively call the merge and count function on the left and right half while keeping update the inversion counts, and merge and count of the left and right.

3. State the recurrence that results from the divide-and-conquer algorithm you designed in Part 3.

$$T(n) = 2T(\frac{n}{2}) + cn$$

4. Solve the recurrence for the divide-and-conquer algorithm using the *substitution method*. For full credit, show your work.

$$T(n) = 2T(\frac{n}{2}) + cn$$

$$T(\frac{n}{2}) = 2T(\frac{n}{4}) + \frac{cn}{2}$$

$$T(\frac{n}{4}) = 2T(\frac{n}{8}) + \frac{cn}{4}$$

$$T(n) = 8T(\frac{n}{8}) + 3cn$$

$$let\ n = 2^k\ then\ \log_2 n = k,\ \ T(n) = 2^k * T\left(\frac{n}{2^k}\right) + k * cn = 2^k + k * cn = n * \log_2 n$$

5. Confirm that your solution to #4 is correct by solving the recurrence for the divide-and- conquer algorithm using the *master theorem*. For full credit, clearly define the values of *a*, *b*, and *c*.

$$a = 2, b = 2, c = 1, since\ \log_b a = \log_2 2 = 1 = c,$$

$$then\ it\ applies\ to\ the\ second\ case\ of\ master\ theorem,$$
$$T(n) = O(nlogn)$$

Team work:

Quicksort worst case: bad pivot picking, since pivot divide the array into two parts, less or larger than the pivot, if one part is empty, all elements besides pivot are in the other part then it’s worst case O(n^2).

Interesting part: compare, analysis of each sorting algorithms: selection sort, insertion sort, merge sort, quick sort and how parallel implementation of quicksort outperformance than merge & merge quicksort.