1 Recursion (Sorting/Selection)

1.1 Counting Inversions

Which of these arrays is more “sorted”?

15	8	23	5	29	12	4

4	8	5	12	23	29	15

Write out a sorted copy of the array in each of the blank tables here. Draw lines from each number in the mixed up list to the corresponding number in the sorted list. What do you notice about these lines? What does that represent?

Define:

An **inversion** in an array $A[1..n]$ is a pair of indices (i, j) such that:

What array of size n has the largest number of inversions, and what is that number? (Assume distinct elements for simplicity.)

What array has the smallest number of inversions?

Brute Force Algorithm

Describe a straightforward, brute-force algorithm to count the number of inversions in an array.

```
procedure CountInversionsBF($A[1..n]$)
    ▷ Input: an array of integers
      Output: the number of pairs of integers in the array that are out of order.
                                                                              ◁
    ▷
                                                                              ◁
```

What is the time complexity of this?

Recursive (Divide-and-conquer)

Describe a recursive algorithm that splits the array into equal halves and recursively finds the inversion counts and then calculates the overall count somehow.

```
procedure CountInversionsRec($A[1..n], i, j$)
    ▷ Input: an array of integers, and indices of a subsection of the array, $A[i..j]$
      Output: the number of pairs of integers in $A[i..j]$ that are out of order.
                                                                              ◁
    ▷
                                                                              ◁
```

What is the time complexity of this?[1]
(Write a recurrence relation and use the Master Theorem.)

[1]What about the space complexity?

Smarter Recursive (Divide-and-conquer)

The trick to making divide-and-conquer efficient is to make it so that conquering is easier than just solving the whole problem.

- Fix some i on the left side. How many j on the right side form inversions?

8	2	91	22	57
$\uparrow$ i				

6	10	1	7	4

- What would we do if the right hand side were sorted?

8	2	91	22	57
$\uparrow$ i				

1	4	6	7	10

- Time?

- So... what's the problem with presorting?

- Ok, sort (both halves of the array) as part of the process...

- Count the total number of inversions between elements in these two halves of an array:

2	8	22	57	91
$\uparrow$ i				

1	4	6	7	10
$\uparrow$ j				

- Does this feel familiar?

Modify merge sort. . .

```
MergeSort($A[1..n]$):
  if $n > 1$
      $m \leftarrow \lfloor n/2 \rfloor$
      MergeSort($A[1..m]$)
      MergeSort($A[m+1..n]$)
      Merge($A[1..n], m$)

Merge($A[1..n], m$):
  $i \leftarrow 1$;  $j \leftarrow m+1$
  for $k \leftarrow 1$ to $n$
      if $j > n$
          $B[k] \leftarrow A[i]$;  $i \leftarrow i+1$
      else if $i > m$
          $B[k] \leftarrow A[j]$;  $j \leftarrow j+1$
      else if $A[i] < A[j]$
          $B[k] \leftarrow A[i]$;  $i \leftarrow i+1$
      else
          $B[k] \leftarrow A[j]$;  $j \leftarrow j+1$
  for $k \leftarrow 1$ to $n$
      $A[k] \leftarrow B[k]$
```

Prove the number of inversions (assume an array of length 2^k) is correctly counted.