Divide-and-Conquer Algorithms

The ***divide and conquer*** strategy solves a problem by :

1. Breaking into ***sub problems*** that are themselves smaller instances of the same type of problem.

2. Recursively solving these sub problems.

3. Appropriately combining their answers.

Merge sort: Merge sort is good for data that's too big to have in memory at once, because its pattern of storage access is very regular. It also uses even fewer comparisons than heap sort, and is especially suited for data stored as linked lists.

- **Merge Sort**

Merge Sort is a O(nlogn) sorting algorithm. It is easy to implement merge sort such that it is stable meaning it preserves the input order of equal elements in the sorted output. It is a comparison sort.

Example:

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	[31	23	01	17	19	28	09	03	13	15	22	08	29]

```
                              1 : 13

          1 : 7                                     8 : 13

   1 : 4           5 : 7                8 : 10               11 : 13

1 : 2     3 : 4     5 : 6    7 : 7     8 : 9    10 : 10   11 : 12   13 : 13

31 23     01 17    19 28     09       03 13      15        22 08       29

23 31     01 17     19 28     09       03 13      15        08 22       29

 01 17 23 31         09 19 28             03 13 15             08 22 29

     01 09 17 19 23 28 31                       03 08 13 15 22 29

                  01 03 08 09 13 15 17 19 22 23 29 31
```

Mergesort pseudocode

```
Algorithm mergesort (A, left, right)
Input :An array A of numbers , the bounds left and
Right for the elements to be sorted
Output :A[left…right] is sorted
If(left<right){  /*we have at least two elements to sort*/
Mid<- [(left +right)/2]
Mergesort (A, left, mid)
/*now A[left….mid]
is sorted*/
mergesort(A,mid+1,right)
/*now A[mid+1….right] is sorted */
Merge(A, left, mid, right)
}
```

- **Complexity of Merge Sort**

Best case:

If the running time of merge sort for a list of length n is $T(n)$, then the recurrence $\boldsymbol{T(n) = 2T(n/2) + n}$ follows from the definition of the algorithm (apply the algorithm to two lists of half the size of the original list, and add the n steps taken to merge the resulting two lists)

Worst and Average case:
In sorting n items, merge sort has an average and worst-case performance of **O(*n* log *n*)**.The worst case,
merge sort does exactly $\boldsymbol{(n \log n - 2^{\log n} + 1)}$ comparisons, which is between $\boldsymbol{(n \log n - n + 1)}$ and $\boldsymbol{(n \log n - 0.9139 \cdot n + 1)}$ [logs are base 2].
Merge sort's ***worst*** case is found simultaneously with quicksort's ***best*** case.
In terms of moves, merge sort's worst case complexity is O(n log n)