Merge Sort → Divide & Conquer

↳ Good for LL

(Inplace)

Heap Sort

↳ clumsy for LL.

but good for arrays.

↳ Non-inplace for arrays.

Quick sort → Best known comparison-based Sorting

↳ D&C

SORT(L, n)

Base: (n == 1) ⇒ L' = L

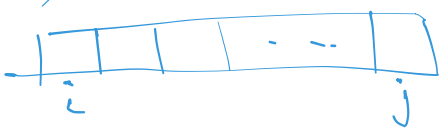
Inductive: Split into two non-empty lists L1 and L2 (Rec.)

split in Middle → MergeSort

$L_1' = SORT(L_1)$

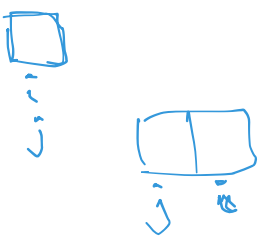
$L_2' = SORT(L_2)$

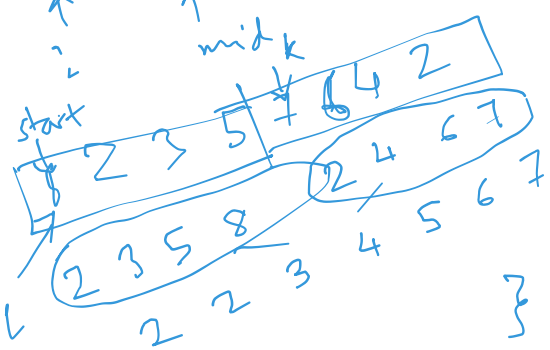
$L' = MERGE(L_1', L_2')$ → Rec.



Algorithm

```
mergeSort(A, i, j)
{
    if (i ≥ j) return;
    mid = (i+j)/2;
    → mergeSort(A, i, mid);
    → mergeSort(A, mid+1, j);
    merge(A, i, j); // Preserves the sorted order
}
```

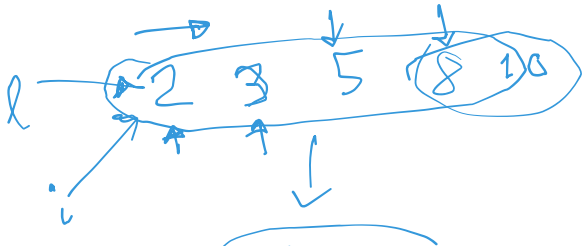




Merge:

merge(A, i, j)

{

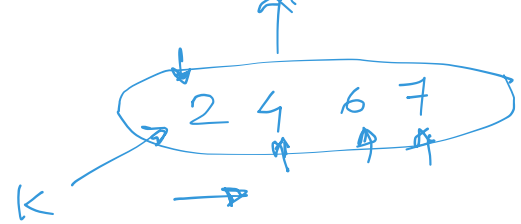


start = i;

mid = (i + j) / 2;

k = mid + 1;

l = i;



B → 2 2 3 4 5 6 7 8 10

Split

```
while (i ≤ mid && k ≤ j) {
    if (A[i] ≤ A[k]) {
        B[l++] = A[i++];
    } else {
        B[l++] = A[k++];
    }
}
if (i > mid) {
    for (; k <= j;) B[l++] = A[k++];
} else {
    for (; i <= mid;) B[l++] = A[i++];
}
```

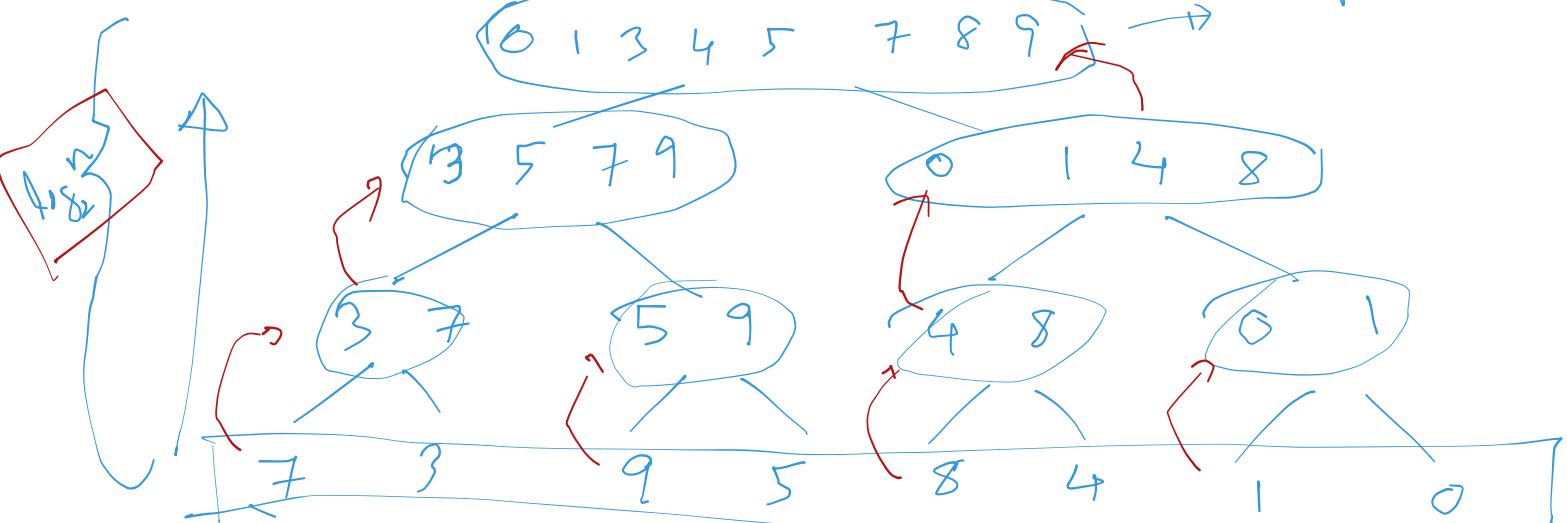
Merge

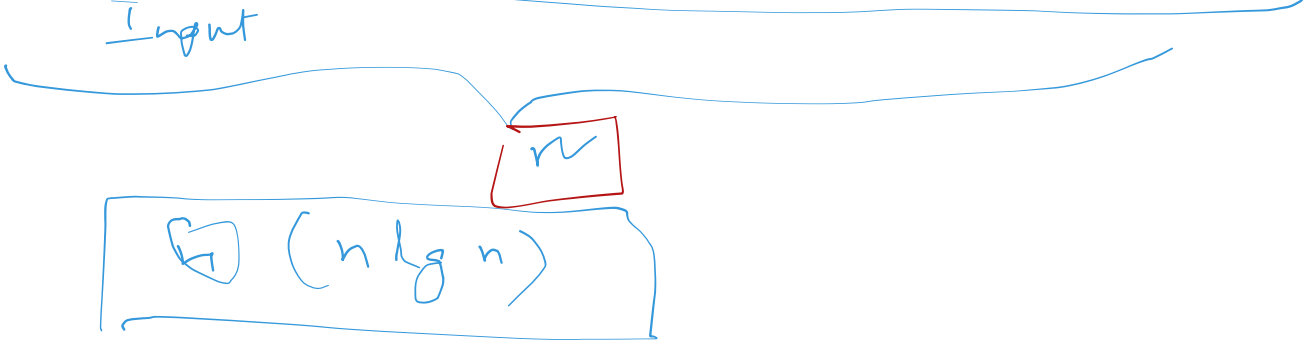
Copy the remaining entries

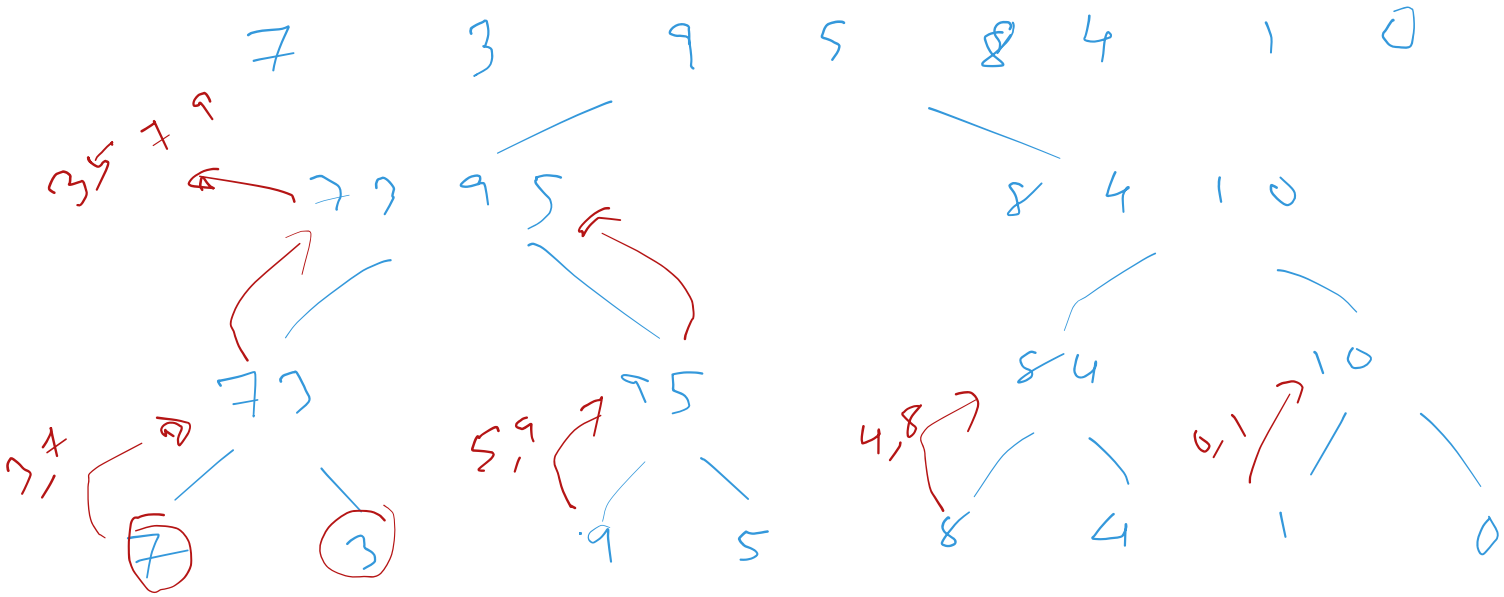
$T(1) = 1$

Exercise

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$







Natural sorting algo for Linked list

→ Merge sort

But Not an inplace sort for Arrays.