Sort

```python
## MergeSort ##
def merge(nums, left, mid, right):
    L, R = nums[left:mid], nums[mid:right+1]
    i = j = 0
    for k in range(left, right+1):
        if j >= len(R) or (i < len(L) and L[i] < R[j]):
            nums[k] = L[i]
            i += 1
        else:
            nums[k] = R[j]
            j += 1

def mergeSort(nums, left, right):
    if left < right:
        mid = left + (right-left+1)//2
        mergeSort(nums, left, mid-1)
        mergeSort(nums, mid, right)
        merge(nums, left, mid, right)

## QuickSort, QuickSelect ##
def partition(nums, left, right):
    wall, pivot = left, right  # pick right as pivot
    for i in range(wall, pivot):
        if nums[i] <= nums[pivot]:
            swap(nums, i, wall)
            wall += 1
    swap(nums, pivot, wall)
    return wall

def quickSort(nums, left, right):
   if left < right:
       wall = partition(nums, left, right)
       quickSort(nums, left, wall-1)
       quickSort(nums, wall+1, right)

def quickSelect(nums, left, right, k):
    if left < right:
        wall = partition(nums, left, right)
        if wall == k-1:  return
        elif wall > k-1:  quickSelect(nums, left, wall-1, k)
        else:  quickSelect(nums, wall+1, right, k)
```

sort

Sort Linked List `merge sort`
Insertion Sort Linked List `hard to code`
Valid Anagram `counter`
Largest Number `sort`
Reorganize String `most common ones`
Intersection of Two Arrays I `counter` `sort, two pointer`
Intersection of Two Arrays II `sort, two pointer`
Sort Transformed Array `抛物线，双指针`

wiggle sort

Wiggle Sort I (a <= b >= c) `贪婪:不对就换, O(n)`
Wiggle Sort II (a < b > c) `找中位数m` `>m:从左奇idx` `<m:从右偶idx` `m:剩余`

quick select 找 k-th / topk / 中位数

hard

Maximum Gap `hard`
Random Pick with Blacklist `hard`
Word Abbreviation `hard`
Best Meeting Point `hard`