1. Majority Element

```
class Solution:
    def majorityElement(self, nums: List[int]) -> int:
        nums.sort()
        n = len(nums)
        return nums[n//2]
```

2. Find Numbers with even number of digits

```
class Solution:
  def findNumbers(self, nums: List[int]) -> int:
    return sum(len(str(x)) % 2 == 0 for x in nums)
```

3. Single Number

```
from typing import List
from collections import defaultdict

class Solution:
    def singleNumber(self, nums: List[int]) -> int:
        n_hash = defaultdict(int)

        for val in nums:
            n_hash[val] += 1

        for val in n_hash:
            if n_hash[val] == 1:
                return val
```

4. Missing Number

```
class Solution:
  def missingNumber(self, nums: list[int]) -> int:
    ans = len(nums)

    for i, num in enumerate(nums):
      ans ^= i ^ num

    return ans
```

5. degree of an array

```
class Solution:
    def findShortestSubArray(self, nums: List[int]) -> int:
        cnt = Counter(nums)
        degree = cnt.most_common()[0][1]
        left, right = {}, {}
        for i, v in enumerate(nums):
            if v not in left:
                left[v] = i
            right[v] = i
        ans = inf
        for v in nums:
            if cnt[v] == degree:
                t = right[v] - left[v] + 1
                if ans > t:
                    ans = t
        return ans
```

6. Largest Number at least twice of others

```
class Solution:
    def dominantIndex(self, nums: List[int]) -> int:
        x, y = nlargest(2, nums)
        return nums.index(x) if x >= 2 * y else -1
```

7. Rotate array

```
def rotate(self, nums: List[int], k: int) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        for i in range(k):
            nums.insert(0,nums.pop())
```

8. Maximum Consecutive Ones

```
class Solution:
    def findMaxConsecutiveOnes(self, nums: List[int]) -> int:
        ans = cnt = 0
        for x in nums:
            if x:
                cnt += 1
                ans = max(ans, cnt)
            else:
                cnt = 0
        return ans
```

9. Remove Duplicate from sorted array

```
class Solution:
    def removeDuplicates(self, nums: List[int]) -> int:
        k = 0
        for x in nums:
```

```
        if k == 0 or x != nums[k - 1]:
            nums[k] = x
            k += 1
    return k
```

10. Maximum Subarray

```
class Solution:
    def maxSubArray(self, nums: List[int]) -> int:
        ans = f = nums[0]
        for x in nums[1:]:
            f = max(f, 0) + x
            ans = max(ans, f)
        return ans
```

11. Find the town judge

```
class Solution:
    def findJudge(self, n: int, trust: List[List[int]]) -> int:
        cnt1 = [0] * (n + 1)
        cnt2 = [0] * (n + 1)
        for a, b in trust:
            cnt1[a] += 1
            cnt2[b] += 1
        for i in range(1, n + 1):
            if cnt1[i] == 0 and cnt2[i] == n - 1:
                return i
        return -1
```

12. Square of an sorted array

```
class Solution:
    def sortedSquares(self, nums: List[int]) -> List[int]:
        ans = []
        i, j = 0, len(nums) - 1
        while i <= j:
            a = nums[i] * nums[i]
            b = nums[j] * nums[j]
            if a > b:
                ans.append(a)
                i += 1
            else:
                ans.append(b)
                j -= 1
        return ans[::-1]
```

13. Find pivot index

```
class Solution:
    def pivotIndex(self, nums: List[int]) -> int:
        left, right = 0, sum(nums)
        for i, x in enumerate(nums):
            right -= x
            if left == right:
                return i
            left += x
        return -1
```

14. Array partition

```
class Solution:
    def arrayPairSum(self, nums: List[int]) -> int:
        nums.sort()
        return sum(nums[::2])
```