LC 461: Hamming Distance

```python
class Solution(object):
    def hammingDistance(self, x, y):
        """
        :type x: int
        :type y: int
        :rtype: int
        """
        return bin(x^y).count('1')
```

Success Details ›

Runtime: 8 ms, faster than 98.20% of Python online submissions for Hamming Distance.

Memory Usage: 11.7 MB, less than 50.00% of Python online submissions for Hamming Distance.

LC 190: Reverse Bits

```python
class Solution:
    # @param n, an integer
    # @return an integer
    def reverseBits(self, n):
        res = 0
        for i in range(32):
            res = (res << 1) + (n & 1)
            n >>= 1
        return res
```

Success Details ›

Runtime: 12 ms, faster than 94.69% of Python online submissions for Reverse Bits.

Memory Usage: 11.6 MB, less than 82.14% of Python online submissions for Reverse Bits.

LC 421: Maximum XOR of Two Numbers in an Array

```python
class Solution(object):
    def findMaximumXOR(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        res = 0
        for i in range(32)[::-1]:
            res <<= 1
            prefixes = {num >> i for num in nums}
            res += any(res^1 ^ p in prefixes for p in prefixes)
        return res
```

Success Details ›

Runtime: 88 ms, faster than 93.93% of Python online submissions for Maximum XOR of Two Numbers in an Array.

Memory Usage: 14.9 MB, less than 100.00% of Python online submissions for Maximum XOR of Two Numbers in an Array.

LC 187: Repeated DNA Sequences

```python
class Solution(object):
    def findRepeatedDnaSequences(self, s):
        """
        :type s: str
        :rtype: List[str]
        """
        str_dic = {}
        res = []
        for i in range(len(s) - 9):
            temp = s[i:i+10]
            if temp in str_dic:
                if str_dic[temp] == 1:
                    res.append(temp)
                str_dic[temp] += 1
            else:
                str_dic[temp] = 1
        return res
```

Success Details ›

Runtime: 44 ms, faster than 95.60% of Python online submissions for Repeated DNA Sequences.

Memory Usage: 33 MB, less than 16.67% of Python online submissions for Repeated DNA Sequences.

LC 201: Bitwise AND of Numbers Range

```python
class Solution(object):
    def rangeBitwiseAnd(self, m, n):
        """
        :type m: int
        :type n: int
        :rtype: int
        """
        i = 0
        while m != n:
            m >>= 1
            n >>= 1
            i += 1
        return m << i
```

Success Details ›

Runtime: 48 ms, faster than 60.60% of Python online submissions for Bitwise AND of Numbers Range.

Memory Usage: 11.8 MB, less than 50.00% of Python online submissions for Bitwise AND of Numbers Range.

LC 1178: Number of Valid Words for Each Puzzle

```python
import collections
class Solution(object):
    def findNumOfValidWords(self, words, puzzles):
        """
        :type words: List[str]
        :type puzzles: List[str]
        :rtype: List[int]
        """
        count = collections.Counter(frozenset(w) for w in words)
        res = []
        for p in puzzles:
            subs = [p[0]]
            for c in p[1:]:
                subs += [s + c for s in subs]
            res.append(sum(count[frozenset(s)] for s in subs))
        return res
```

Success Details ›

Runtime: 856 ms, faster than 68.89% of Python online submissions for Number of Valid Words for Each Puzzle.

Memory Usage: 68.4 MB, less than 100.00% of Python online submissions for Number of Valid Words for Each Puzzle.