LC 401: Binary Watch

```python
class Solution(object):
    def readBinaryWatch(self, num):
        """
        :type num: int
        :rtype: List[str]
        """
        res = []
        if num == 0:
            return ["0:00"]
        def backtrack(hour, minute, n, start):
            if hour >= 12 or minute >= 60:
                return
            if n == 0:
                res.append('%d:%02d' % (hour, minute))
                return
            for i in range(start, 10):
                if i < 4:
                    backtrack(hour + 2**i, minute, n - 1, i + 1)
                else:
                    backtrack(hour, minute + 2**(i - 4), n - 1, i + 1)
        backtrack(0, 0, num, 0)
        return res
```

Success Details ›

Runtime: 8 ms, faster than 99.84% of Python online submissions for Binary Watch.

Memory Usage: 11.8 MB, less than 33.33% of Python online submissions for Binary Watch.

LC 39: Combination Sum

```
class Solution(object):
    def combinationSum(self, candidates, target):
        """
        :type candidates: List[int]
        :type target: int
        :rtype: List[List[int]]
        """
        result = []
        def find_residual(result, temp_result, candidates, max_ele, residual):
            for item in candidates:
                if item < max_ele:
                    continue
                if item > residual:
                    return
                if item == residual:
                    residual = 0
                    temp_result.append(item)
                    result.append(temp_result)
                    return
                if item < residual:
                    #a = [i for i in temp_result]
                    #a.append(item)
                    find_residual(result, temp_result+[item], candidates, item, residual-item)
        candidates.sort()
        find_residual(result, [], candidates, 0, target)
        return result
```

Success Details ›

Runtime: 40 ms, faster than 88.70% of Python online submissions for Combination Sum.

Memory Usage: 11.7 MB, less than 79.59% of Python online submissions for Combination Sum.

LC 46: Permutations

```python
class Solution(object):
    def permute(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        if not nums :
            return []
        res = []
        self.permute_dfs(nums, [], res)
        return res

    def permute_dfs(self, nums, path, res):
        if not nums:
            res.append(path)
            return
        for i in range(len(nums)):
            new_nums = nums[:i] + nums[i+1:]
            self.permute_dfs(new_nums, path+[nums[i]], res)
```

Success Details >

Runtime: 20 ms, faster than 96.67% of Python online submissions for Permutations.

Memory Usage: 11.7 MB, less than 98.00% of Python online submissions for Permutations.

LC Permutations II

```
class Solution(object):
    def permuteUnique(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        if not nums:
            return []
        nums.sort()
        res = []
        self.dfs(nums, [], res)
        return res

    def dfs(self, nums, path, res):
        if not nums:
            res.append(path)
            return
        for i in range(len(nums)):
            if i > 0 and nums[i] == nums[i-1]:
                continue
            new_nums = nums[:i] + nums[i+1:]
            self.dfs(new_nums, path+[nums[i]], res)
```

Success Details ›

Runtime: 32 ms, faster than 97.63% of Python online submissions for Permutations II.

Memory Usage: 11.9 MB, less than 66.67% of Python online submissions for Permutations II.

LC 52: N-Queens II

```
class Solution(object):
    def totalNQueens(self, n):
        """
        :type n: int
        :rtype: int
        """
        self.res = 0
        cols = [-1]*n

        def check_valid(locs, row, col):
            for i in range(0, row):
                if col == locs[i] or row - i == abs(col - locs[i]):
                    return False
            return True

        def solu(cols, row):
            if row == n:
                self.res += 1
                return
            else:
                for col in range(n):
                    cols[row] = col
                    if check_valid(cols, row, col):
                        solu(cols, row + 1)
        solu(cols, 0)
        return self.res
```

Success Details ›

Runtime: 72 ms, faster than 53.72% of Python online submissions for N-Queens II.

Memory Usage: 11.9 MB, less than 25.00% of Python online submissions for N-Queens II.

Implement a queue class given a stack class

```python
class Stack_To_Queue(object):
    def __init__(self, stack):
        self.stack1 = [i for i in stack]
        self.stack2 = []

    def add(self, num):
        self.stack1.append(num)

    def pop(self):
        if self.stack2:
            return self.stack2.pop()

        else:
            self.stack2 = [self.stack1.pop() for i in range(len(self.stack1))]
            self.stack1 = []
            if self.stack2:
                return self.stack2.pop()
            else:
                print('No Elements in Queue!!!')

stack = [1, 2, 3, 4, 5]
queue = Stack_To_Queue(stack)
print(queue.pop())
print(queue.pop())
queue.add(6)
print(queue.pop())
print(queue.pop())
print(queue.pop())
queue.add(7)
print(queue.pop())
print(queue.pop())
print(queue.pop())
```

```
2
3
4
5
6
7
No Elements in Queue!!!
None
```