452. Minimum Number of Arrows to Burst Balloons

```
class Solution:
    def findMinArrowShots(self, points: List[List[int]]) -> int:
        if not points or not points[0]:
            return 0
        points.sort()
        cur_s, cur_e = points[0]
        count = 1
        for s, e in points[1:]:
            if s <= cur_e:
                cur_e = min(cur_e, e)
            else:
                count += 1
                cur_e = e
        return count
```

986. Interval List Intersections

We will use two pointers and TC is O(m + n), SC is O(max(m, n))

```
class Solution:
    def intervalIntersection(self, A: List[List[int]], B: List[List[int]]) -> List[List[int]]:
        length_A = len(A)
        length_B = len(B)
        i, j = 0, 0
        result = []
        while i < length_A and j < length_B:
            s_A, e_A = A[i]
            s_B, e_B = B[j]
            if not (s_A > e_B or s_B > e_A):
                result.append([max(s_A, s_B), min(e_A, e_B)])

            if e_B > e_A:
                i += 1
            elif e_B < e_A:
                j += 1
            else:
                i += 1
                j += 1
        return result
```

438. Find All Anagrams in a String

```
from collections import defaultdict
class Solution:
    def findAnagrams(self, s: str, p: str) -> List[int]:
```

```
        mem = defaultdict(int)
        l, r = 0, 0
        counter = 0
        result = []
        for c in p:
            if c not in mem:
                counter += 1
            mem[c] += 1
        while r < len(s):
            if s[r] not in mem:
                r += 1
                while l < r:
                    if s[l] in mem:
                        if mem[s[l]] == 0:
                            counter += 1
                        mem[s[l]] += 1
                    l += 1
            else:
                if mem[s[r]] > 0:
                    mem[s[r]] -= 1
                    if mem[s[r]] == 0:
                        counter -= 1
                        if counter == 0:
                            result.append(l)
                else:
                    while l < r:
                        if s[r] == s[l]:
                            l += 1
                            if counter == 0:
                                result.append(l)
                            break

                        if mem[s[l]] == 0:
                            counter += 1
                        mem[s[l]] += 1
                        l += 1
                    r += 1
        return result
```

Simple version:

```
from collections import defaultdict
class Solution:
    def findAnagrams(self, s: str, p: str) -> List[int]:
```

```
        mem = defaultdict(int)
        length = len(p)
        l, r = 0, 0
        counter = 0
        result = []
        for c in p:
            if c not in mem:
                counter += 1
            mem[c] += 1

        while r < len(s):
            mem[s[r]] -= 1
            if mem[s[r]] == 0:
                counter -= 1
            r += 1

            while counter == 0:
                if r - l == length:
                    result.append(l)
                if s[l] in mem:
                    if mem[s[l]] == 0:
                        counter += 1
                    mem[s[l]] += 1
                l += 1
        return result
```

567. Permutation in String

```
from collections import defaultdict
class Solution:
    def checkInclusion(self, s1: str, s2: str) -> bool:
        mem = defaultdict(int)
        length = len(s1)
        l, r = 0, 0
        counter = 0
        for c in s1:
            if c not in mem:
                counter += 1
            mem[c] += 1

        while r < len(s2):
            mem[s2[r]] -= 1
            if mem[s2[r]] == 0:
                counter -= 1
```

```
            r += 1

            while counter == 0:
                if r - l == length:
                    return True
                if s2[l] in mem:
                    if mem[s2[l]] == 0:
                        counter += 1
                    mem[s2[l]] += 1
                l += 1
        return False
```

242. Valid Anagram

```
from collections import Counter
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        return Counter(s) == Counter(t)
```